## REMARKS/ARGUMENTS

Claims 1-32 remain in this application as originally filed.

In the office action dated June 23, 2003, the Examiner rejected all 32 pending claims. Claims 23-32 were rejected under 35 U.S.C. § 102(e) as being anticipated by *Hickman et al.*, U.S. Patent No. 6,523,036. Claims 1-22 and 31 were rejected under 35 U.S.C. § 103 as being unpatentable over *Hickman et al.* in view of *Berkowitz et al.,* U.S. Patent No. 6,457,021. The applicant acknowledges with appreciation, the Examiner's detailed description of the manner in which the references were applied to the claims. However, the cited references do not appear to anticipate or render any of the claims obvious.

*Hickman et al.* does not disclose a database wherein records are subdivided so that static data is stored in static memory devices and dynamic data is stored in dynamic memory devices. At least some of these limitations, either alone or in combination, are found in independent claims 1, 12, 23, 25, 27 and 32. *Hickman et al.* does not distinguish between the type of data stored in its netstore database (the only type of data stored is apparently I-data) and while the data is stored in multiple locations, i.e. on a plurality of clusters of servers, there is no indication that the storage devices are distinguishable in any way or, if such a distinction does exist that data is stored based on that distinction and some nature of the data. *Hickman et al.* discloses using hash functions to distribute the data. Those skilled in the art recognize that hash functions of the type disclosed in *Hickman et al.* facilitate even distribution of the hashed material not distribution of the hashed material based on an inherent quality of the material. However, rather than rely on this characterization of the disclosure of *Hickman et al.*, all that is

necessary is to examine the language of *Hickman et al.* cited by the Examiner and definitions of terms contained therein to establish that *Hickman et al.* does not disclose each and every element and limitation of claims 23-32, as is required to support an anticipation rejection of those claims.

It is well settled law that a prima facie case of anticipation of a claim by a reference requires that the reference disclose, either literally or inherently, every element and limitation of the claim. In attempting to support the anticipation rejections the Examiner has quoted the claim language of each claim rejected as being anticipated and has provided citations to sections of *Hickman et al.* for **some** of the elements and limitations. While the Examiner is under no obligation to specifically point to sections of references establishing the disclosure of every element of the rejected claim, the reference must contain such disclosure, either literally or inherently.

The Examiner asserts that:

> As to claim 23, Hickman et al. discloses a method for creating a database, said method comprising the steps of:
> storing a set of static data elements in a static memory device (see column 7, lines 12-27); and,
> storing a set of dynamic data (i.e. I-data) elements in a dynamic memory device (i.e. netstore) (see column 6, lines 41-49), wherein said database (20) comprises said static data elements and said dynamic data (i.e. I-data) elements (see column 11, lines 40-65; also see column 6, lines 41-49). *First Office Action* ("*FOA*"), pg. 2.

The most that can be said about *Hickman et al.* is that it recognizes that I-data, as that term is defined therein (which definition is set forth hereunder), is "more dynamic" than other types of data stored in typical databases. *Hickman et al.* does not distinguish between static data and dynamic data and does not disclose a database

wherein static data is stored in a static memory device and dynamic data is stored in a dynamic memory device, as recited in claim 23.

The Examiner's assertion that netstore is a dynamic memory device is not supported by *Hickman et al.* The Examiner tries to equate netstore to a dynamic memory device. *Hickman et al.* does not so limit "netstore", instead *Hickman et al.* provides that:

> The term "netstore" as used herein is defined to be an Internet-scale data store that can handle both the traffic and capacity required by an Internet application. The netstore must have several capabilities. First, the typical number of total users that can access the netstore is extremely large (e.g., greater than 100 million users). Additionally, the typical number of concurrent users is large (e.g., 1 million users). Read operations to the netstore are more prevalent than write operations (e.g., a 10-1 read-to-write ratio for some Internet applications or even 100-1 for others). The netstore must be able to store a large amount of data and should be simple and flexible. Additionally, the data stored therein can be treated as a collection of picks that only has meaning to the particular Internet application. *Hickman et al.*, Col. 1, ll. 27-42.

This definition does not establish that the netstore is a "dynamic memory device" as that term is used in claim 23. Even if netstore could be established to be a "dynamic memory device" no candidate for a "static memory device" is identified in *Hickman et al.*

In fact *Hickman et al.* discloses that :

> The system architecture implements a netstore as a set of cooperating server machines. This set is divided into clusters, each of which consists of one or more server machines. All machines within a cluster are replicas of one another and store the same data records. The data is partitioned among the clusters, so that each data record in the netstore is stored in exactly one cluster. *Hickman et al.*, Col. 2, ll. 42-48.

Thus, the netstore disclosed in *Hickman et al.* is not a separate form of distinct types of storage but rather a group of server machines among which data is split up for storage. The data is split along data record lines and individual data records are stored on all of

the machines of a cluster. The data within a data record is not split up to determine if it is static or dynamic data and then stored in separate locations according to its classification but rather all of the data in a data record is stored in multiple copies on the multiple machines forming a single cluster in *Hickman et al.*

The Examiner relies on the following language to establish that *Hickman et al.* discloses "storing a set of static data elements in a static memory device":

> The present invention comprises a netstore architecture for I-data that exhibits high scalability. As will be described in greater detail below, the present invention provides a netstore that has several key features. The first is adaptive partitioning, which refers to the capability to automatically determine the best location to host data based on the real-time state of the netstore, using parameters such as number of clusters, system load and data distribution. The netstore adapts to the current state and updates its partitioning algorithm accordingly. This is in contrast to static partitioning, wherein the partitioning algorithm must be defined by the user during development, and cannot be changed after the system is deployed. Static partitioning is not pragmatic, since there usually is not enough information known at design time to select an efficient static partitioning scheme. *Hickman et al.*, Col. 7, ll. 12-27.

The above quotation, relied upon by the Examiner to establish the "storing a set of static data elements in a static storage device" limitation of claim 23, only establishes that some set of data is stored somewhere. While the quoted language uses the term "static" it is used with regard to "static partitioning" (defined in the quote as partitioning in accordance with a predetermined scheme) not with regard to "static data" or "a static storage device." *Hickman et al.* partitions data for storage in different location in the netstore based on such things as "the best location to host data based on the real-time state of the netstore, using parameters such as number of clusters, system load and data distribution." Thus the location of data storage in *Hickman et al.* is not determined based on the nature of the data or the nature of the storage location as recited in claim 23.

The Examiner relies on Column 11, lines 40-65 and column 6 lines 41-49 for the proposition that the database in *Hickman et al.* "comprises said static data elements and said dynamic data elements." Those cited sections of *Hickman et al.* state that:

> A key aspect to the present invention is the ability to adjust partitioning of a database(s) on the fly, without having to shut down the servers and re-architect the database(s). In typical large-scale databases, tables are partitioned according to a predefined (i.e., fixed) scheme. Partitioning databases in this manner generally yields improved performance, but does not solve the problem associated with large increases in the volume of data transactions typical of growing enterprises. The ultimate benefit provided by adaptive partitioning is that transaction throughput can be increased by merely adding more resources (i.e., servers and support software) without having to re-engineer the system.
>
> Adaptive partitioning is based on breaking the database tables into fragmented subsets, and mapping individual data objects to such fragments based on hashing techniques. In general, hashing divides data objects (i.e., table records) among a number of units of storage, commonly referred to as buckets, based on a hashing function that is typically based on a search-key value of each record.
>
> With reference to FIG. 4, suppose there is an exemplary hashing scheme that distributes data objects in an RDBMS database 63 among four buckets labeled 0-3 using a hashing function, $h(K_i)$, wherein $K_i$ represents a search key corresponding to a respective hash bucket. Ideally, the data objects should be distributed among the buckets such that the distribution is uniform. *Hickman et al.*, Col 11, ll. 40-65
>
> The term "I-data" refers to Internet data. Most of the data that is stored and manipulated by Internet applications does not fit a conventional relational or object-oriented data model well. In general, Internet application data is much simpler and more dynamic than the data designed to fit the relational and object oriented data models. Typically, I-data is also simple and flat. Data design is minimal and is expressed easily through extensible markup language (XML). *Hickman et al.*, Col. 6, ll. 41-49.

The first quotation establishes that the system contemplated by *Hickman et al.* divides

data in a database into hash buckets using hash functions with the goal of even distribution of the data, this does not inherently disclose storing static data in a static memory device and storing dynamic data in a dynamic memory device as required by claim 23. The statement that "Internet application data is much simpler and more dynamic than the data designed to fit the relational and object oriented data models" does not establish that a database in *Hickman et al.* stores a set of static data in a static memory device and a set of dynamic data in a dynamic memory device as required by claim 23. If data other than the "more dynamic" internet application data is stored in a database, this less "dynamic data" could be stored in other traditional databases (i.e. relational and object oriented databases managed by DBMS (see Col. 1, lines 42-65) not in the netstore database based on RDBMS disclosed in *Hickman et al.* This non-internet data need not be "static data" as recited in claim 23.

The Examiner appears to believe that the "more dynamic" internet data (I-data) satisfies the dynamic data recitation of claim 23. Assuming, without admitting, that this is true, there is no indication that the netstore satisfies the dynamic storage device recitation of claim 23. *Hickman et al.* does not disclose, either literally or inherently, all of the elements and limitations of claim 23. Thus, claim 23 is not anticipated by *Hickman et al.* and the rejection should be withdrawn.

Claim 24 depends from claim 23 which was not anticipated by *Hickman et al.* The additional language in *Hickman et al.* cited by the Examiner to reject claim 24, instead of supporting the Examiners position, in fact helps to establish that neither claim 23 nor claim 24 are anticipated.

The relevant cited language in *Hickman et al.* states:

> The term "database" is defined as a set of files that store a collection of data. For example, in an Oracle database, these files include data files, control files, index files, stored procedure files, etc. The data in a relational database is stored in tables comprising columns and rows, wherein each row includes multiple fields and the data in a given row is referred to as a record or record object. The database also includes a set of metadata that is used to define the schema (i.e., structure) of the database; including but not limited to the names of the tables and columns, the types and sizes of the columns, and indexing and storage specifications. *Hickman et al.*, Col 5, ll. 44-55

As the quoted language establishes that, even if the term "metadata" is read on "catalog", *Hickman et al.* does not disclose that the catalog specify "that at least some of said data fields are stored in said static memory device and that at least some of said data fields are stored in said dynamic memory device" as required by claim 24. Not only does *Hickman et al.* not disclose all of the elements and limitations of claim 23 from which claim 24 depends, it does not disclose all of the additional limitations recited in claim 24. *Hickman et al.* does not anticipate claim 24.

The Examiner asserts that:

> As to claims 25 and 30, Hickman et al. discloses a method for editing a data element stored in a static memory device comprising a plurality of storage units, said method comprising the steps of:
> copying a content of one of said storage units to a dynamic memory device (i.e. netstore), wherein said content comprises said data element (see column 9, lines 1-26; where "copying" is read on "read or write or update");
> editing (i.e. modifying/updating) said data element while said data element is stored in said dynamic memory;
> erasing said one of said storage units (see column 9, lines 1-26; where "erasing" is read on "deleting");
> and, writing said content, including said data element that has been edited, into said one of said storage units (see column 11, lines 7-18). *FOA*, pg. 3.

The entire storage unit for the database disclosed in *Hickman et al.* is the netstore which is not identified as a dynamic memory device. Even if the netstore is a dynamic memory device, claim 25 requires that the information be copied from static memory to dynamic memory. *Hickman et al.* does not distinguish between different areas of storage in the netstore based on the type of memory device. *Hickman et al.* does not disclose all of the elements and limitations of claims 25 and 32 and thus the rejections of those claims should be withdrawn.

While the cited language may establish that *Hickman et al.* copies data resident in the database stored on the netserve and writes or updates that data to all of the bases in a particular cluster, that is not what claims 25 and 30 recite. *Hickman et al.* nowhere teaches or suggests copying static data from a storage unit in a static memory device to dynamic memory, editing the copied data while it is in dynamic memory, erasing the storage unit and writing the edited data back into the erased storage unit as recited in claims 25 and 30.

As to claims 26 and 32, the Examiner asserted that:

> Hickman et al. discloses a method performed by a database generation tool for creating a compressed database, said method comprising the steps of:
>
> receiving a data input file, said data input file defining a first set of data fields to be included in said database and said data input file including a set of data elements to be included in said database (see column 26, line 66 through column 27, line 5);
>
> identifying a second set of data fields in said data input file that are designated to contain a Boolean element, said second set of data fields being a subset of said first set of data fields (see column 29, lines 15-21; where "second set of data fields" is read on "adding new base");
>
> defining one or more new data fields for collectively storing said Boolean elements (see column 19, lines 56-65; where "Boolean elements" is read on "types specified in the schema");

> modifying said first set of data fields to eliminate said second set of data fields (see column 22, lines 43-48; where "eliminating" is read on "deleting"); and,
>
> generating a catalog that defines an arrangement of said first set of data fields (see column 5, lines 44-55; where "catalog" is read on "metadata") wherein said arrangement includes said one or more new data fields for collectively storing said Boolean elements(see column 19, lines 56-65; where "Boolean elements" is read on "types specified in the schema"). *FOA*, pgs. 3-4.

Quite simply, *Hickman et al.* discloses nothing even similar to what is claimed in claims 26 and 32. These claims recite restructuring the data record in the event that Boolean elements are present to collect the Boolean elements into fewer data fields to reduce the number of data fields and thereby compress the database. In the non-limiting example of the application, the data fields tend to be word size, while Boolean elements are bit size. Thus memory is wasted if separate data fields are used to store each Boolean element in a record.

The language relied upon by the Examiner for rejecting these claims establishes that, rather than saving storage space by compressing data through accumulation of Boolean data into a new field, *Hickman et al.* in fact creates multiple copies of data thus increasing the usage of storage space. The cited language that has not been previously quoted above states:

> b. System tables: The entire contents of the system tables are read out of $M_1$ to a file using Sequel Server's BCP application programming interface. They are then read in to $M_5$ and $M_6$.
>
> c. User tables: each user table is created on $M_5$ and $M_6$ in case it does not exist already, using the create statement stored in the create column of $M_1$'s metatable. *Hickman et al.*, Col. 26, l. 66 - Col. 27, l. 5.
>
> The addition of new base $M_7$ (i.e., the new node) is graphically depicted in FIG. 11. Note that the addition of a new node does not alter the fragment

> map, but rather alters the cluster map, which is broadcast to the API's in the system to enabling load balancing across all of the nodes in the cluster to which the new node is added (i.e., cluster $C_1$ in this instance). *Hickman et al.*, Col. 29, ll. 15-21.
>
> An object in the system is a structure consisting of fields of predefined type. These types are specified in the schema in which the object will reside. Every object has a primary key that uniquely determines the object. A user specifies the values for each of the fields. This can be done in many ways; an XML representation of that value assignment is presently supported, although other representations could be implemented as well. *Hickman et al.*, Col. 19, ll. 56-65.
>
> a. Change the data's row status from "C" to "D", indicating that a delete is in progress but has not committed. Any retrieve operations make no distinction between "C" and "D", but no other operation that would change the data may begin.
>
> b. A row is added to the open transaction table. *Hickman et al.*, Col. 22, ll. 43-48.

The quoted language, relied upon by the Examiner to establish compression, clearly shows that *Hickman et al.* increases memory usage by storing multiple copies of the data within the database and does not compress the data in the manner recited in claims 26 and 32. *Hickman et al.* does not contain all of the elements and limitations recited in claims 26 and 32 and thus does not anticipate those claims.

The Examiner asserted that:

> As to claim 27, Hickman et al. discloses a computer program product comprising a computer readable code stored on a computer readable medium, that when executed, causes a computer to:
> read a catalog (i.e. metadata) to determine where a set of static data shall be stored in a static memory device (see column 5, lines 44-55; where "catalog" is read on "metadata");
> store said static data in said static memory device according to said catalog (see column 3, lines 17-25);
> read said catalog to determine where a set of dynamic data (i.e. I-data) shall be stored in a dynamic memory device (i.e. netstore) (see column 3, lines 47-62); and

store said dynamic data (I-data) in said dynamic memory device according to said catalog (see column 1, lines 29-32; also see column 2, lines 42-48). *FOA*, pgs. 4-5.

The Examiner also asserted that:

As to claim 28, Hickman et al. discloses a method, said computer program product further causing said computer to:

store said static data as a static data file in said static memory device (see column 3, lines 17-25); and

store said dynamic data (i.e. I-data) as a dynamic data file in said dynamic memory device (i.e. netstore)(see column 1, lines 29-32; also see column 2, lines 42-48). *FOA*, pg. 5.

The Examiner also asserted that :

As to claim 29, Hickman et al. **as modified** discloses a system further comprising a file system adopted to access said dynamic data (i.e. I-data) contained in said dynamic data (i.e. I-data) file using one or more memory pointers (see Table 12; where pointers is read on "IP addresses"). *FOA*, pg. 5.

As stated above, *Hickman et al.* does not distinguish between static data and dynamic data, does not distinguish between static memory devices and dynamic memory devices, does not store data in one or the other type of memory device based on the memory type and does not indicate that the metadata, if it is considered to be a catalog, contains information for distinguishing static data from dynamic data. *Hickman et al.* does not anticipate claim 27, or any claim depending therefrom, because *Hickman et al.* does not disclose all of the elements and limitations of claim 27.

Claims 1-22 and claim 31 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Hickman et al.* (U.S. Patent No. 6,523,036) in view of *Berkowitz et al.* (U.S. Patent No. 6,457,021). Of these claims, claims 1 and 12 are independent claims and claim 2-11, 13-22 and 31 are independent claims. The Examiner asserts that:

> As to claim 1 Hickman et al. discloses a database 20 partitioned into a first section and a second section (see column 8, lines 28-32), the first section comprising static data and being stored in a static memory device (see column 7, lines 11-27), the second section comprising dynamic data (i.e. I-data) and being stored in a dynamic memory device (i.e. netstore) (see column 6, lines 41-49).
>
> Hickman et al. does not disclose a data base manager for managing said database.
>
> Berkowitz et al. teaches a database manager for managing said database (see Fig. 2, element 201; also see column 5, lines 24-30).
>
> Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Hickman et al. to include a database manager for managing said database. *FOA*, pg. 6.

Whether a database manager exists in *Hickman et al.* (the RDBMS may qualify) or *Berkowitz et al.* and whether there is any teaching or suggestion in the references to combine the teachings thereof is moot, since, as explained above, *Hickman et al.* does not distinguish between data based on whether it is static or dynamic and does not include static and dynamic storage devices as recited in claim 1. The additional citation by the Examiner of the language of column 8, lines 28-32, ("Although depicted as a separate layer in the Figure, it will be understood that virtual transaction layer 28 also includes components that reside in both application server layer 18 and on machines on which partitions of databases 20 are stored.") does not cure the deficiency established above.

The combination of *Hickman et al.* and *Berkowitz et al.* does not disclose every element and limitation of claim 1. As stated above, *Hickman et al.* does not disclose the creation of a database with static data stored in a static memory device and dynamic data stored in a dynamic memory device as required by claim 1. The disclosure

of *Berkowitz et al.* does not remedy this shortcoming in *Hickman et al.* Thus the obviousness rejection of claim 1 is improper. Claims 2-11 depend from claim 1. Since the combination of *Hickman et al.* and *Berkowitz et al.* does not disclose all of the elements and limitations of claim 1, claims 2-11 are not rendered obvious by such combination. Applicant reserves the right to assert additional bases for establishing the patentability of claims 1-11 over the combination of *Hickman et al.* and *Berkowitz et al.*, including, but not limited to, the lack of disclosure by the combination of additional elements and limitations of the claims, whether there is a teaching, motivation, or suggestion to select and combine the references relied on as evidence of obviousness.

The Examiner asserts that:

> As to claim 12, Hickman et al. discloses a control system having a data storage system for storing data related to said control system, the control system comprising:
> a communication network (see Fig. 14);
> an application node coupled to said communication network, the (*sic.*) application node (see Fig. 2) having a static memory device and a dynamic memory device (i.e. netstore)(See column 8, lines 20-38);
> a database partitioned into a first section and a second section (see column 8, lines 28-32), said first section comprising static data and being stored in said static memory device, said second section comprising dynamic data (i.e. I-data) and being stored in said dynamic memory device (i.e. netstore)(see column 6, lines 41-49; also see column 7, lines 12-27); and (sic.) disposed in said application node (32) for managing said database see Fig. 2; also see column 8, lines 20-38).
>
> Hickman et al. does not teach the database manager.
>
> Berkowitz et al. teaches a database manager (see Fig. 2, element 201; also see column 5, lines 24-30).
>
> Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Hickman et al. to include a database manager. *FOA*, pgs. 10-11.

Again, the combination of *Hickman et al.* and *Berkowitz et al.* does not disclose every element and limitation of claim 12. As stated above, *Hickman et al.* does not disclose the creation of a database with static data stored in a static memory device and dynamic data stored in a dynamic memory device as required by claim 12. The disclosure of *Berkowitz et al.* does not remedy this shortcoming in *Hickman et al.* Thus the obviousness rejection of claim 12 is improper. Claims 13-22 depend from claim 12. Since the combination of *Hickman et al.* and *Berkowitz et al.* does not disclose all of the elements and limitations of claim 12, claims 13-22 are not rendered obvious by such combination. Once again applicant reserves the right to assert additional bases for establishing the patentability of claims 12-22 over the combination of *Hickman et al.* and *Berkowitz et al.*, including, but not limited to, the lack of disclosure by the combination of additional elements and limitations of the claims, whether there is a teaching, motivation, or suggestion to select and combine the references relied on as evidence of obviousness.

Claim 31 depends from claim 27. As stated above, claim 27 is not anticipated by *Hickman et al.* because *Hickman et al.* does not disclose every element and limitation of claim 27. The elements and limitations of claim 27 that are not taught by *Hickman et al.* are not taught in *Berkowitz et al.* either. Thus, a prima facie case of obviousness of claim 31 cannot be established as the combination of references does not teach or suggest every element and limitation of claim 27 from which claim 31 depends. Applicant reserves the right to assert additional bases for establishing the patentability of claim 31 over the combination of *Hickman et al.* and *Berkowitz et al.*, including, but not limited to, the lack of disclosure by the combination of additional elements and

limitations recited in either claim 27 or claim 31 and whether there is a teaching, motivation, or suggestion to select and combine the references relied on as evidence of obviousness.

Neither *Hickman et al.* nor *Berkowitz et al.*, alone or in combination, teach or suggest every element and limitation recited in any of the thirty-two pending claims. Thus, claims 1-32 are in condition for allowance. Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,
MAGINOT, MOORE & BECK

By _____
David B. Quick
Reg. No. 31,993
Tel.:(317) 638-2922